**Full Stack Take Home Test**

1. In Angular JS When a scope is terminated, two similar “destroy” events are fired. What are they used for, and why are there two?

(Ans). In Angular JS when a scope is terminated, two similar destroy events are triggere.

The explanation of each “destroy” event are mentioned below

1. **$destroy**, is an Angular JS event, It can be used by Angular JS scopes

Where they are accessible, e.g. in controllers.

1. **$destroy**. is a jQuery event, It is triggered whenever a node is removed
2. What is the preferred method of resolving unhandled exceptions in Node.js?

(Ans).

1. What is typically the first argument passed to a Node.js callback handler?

(Ans). **Error object** is the first argument passed to Node.js callback

1. List at least three ways to communicate between modules of your application using core AngularJS functionality.

(Ans). To communicate between modules of our application using core AngularJS

functionality, The below mentioned three ways can be used :

1) Using Events

2) Using Services

3) By assigning models on $rootScope

1. Write a MongoDb query to retrieve all the documents in Users collection the status as the status equals "A" and age is less than ($lt) 30:

(Ans). db.Users.find({ $and : [ { “status” : “A” } , { “age” : { $lt:30 } } ] }).pretty()

1. Consider following code snippet:

{  
 console.time("loop");  
 for (var i = 0; i < 1000000; i += 1){  
 // Do nothing  
 }  
 console.timeEnd("loop");  
 }

The time required to run this code in Google Chrome is considerably more than the time required to run it in Node.js. Explain why this is so, even though both use the v8

JavaScript Engine.

(Ans). Within a Chrome, declaring the variable i is outside of any function’s scope makes

it global and therefore it binds it as a property of the window object. So, running this

code in a Chrome requires repeatedly resolving the property i within the

heavily populated window namespace in each iteration of the for loop.

In Node.js, declaring any variable outside of any function’s scope binds it only

to the module’s own scope which makes it much easier and faster to resolve.

7. This is a simple test written for Protractor (A test framework for Angular JS), a slightly modified example from Protractor docs:

it('should find an element by text input model', function() {

browser.get('/some-url');

var login = element(by.model('username'));

login.clear();

login.sendKeys('Jane Doe');

var name = element(by.binding('username'));

expect(name.getText()).toEqual('Jane Doe');

// Point A

});

Explain if the code is synchronous or asynchronous and how it works.

8. Given two tables created and populated as follows:

CREATE TABLE dbo.envelope(id int, user\_id int);

CREATE TABLE dbo.docs(idnum int, pageseq int, doctext varchar(100));

INSERT INTO dbo.envelope VALUES

(1,1),

(2,2),

(3,3);

INSERT INTO dbo.docs(idnum,pageseq) VALUES

(1,5),

(2,6),

(Null,0);

What will the result be from the following query:

UPDATE docs SET doctext=pageseq FROM docs INNER JOIN envelope ON envelope.id=docs.idnum

WHERE EXISTS (

SELECT 1 FROM dbo.docs

WHERE id=envelope.id

);

(Ans). The result of the query is following table

|  |  |  |
| --- | --- | --- |
| idnum | pageseq | doctext |
| 1 | 5 | 5 |
| 2 | 6 | 6 |
| Null | 0 | Null |

9. Go to [jsfiddle.net](http://jsfiddle.net/), and paste in the markup given below into the HTML area.

<div class="grader">GRADER</div>  
<div class="panda">panda</div>

Now enter just enough CSS (no HTML or JS!) to make the output look exactly like the following:



Hit 'Save' in the JSFiddle toolbar, which will redirect you to a unique URL for your work. Paste in that URL below .

(Ans). <https://jsfiddle.net/3t0zctxt/>

10. What is REST?

11. What is HTTP Strict Transport Security (HSTS)?

12. **Why won't this Node program run? What's wrong with it?**

var express = require('express');  
var app = express.createServer(express.logger());  
  
app.get('/', function(req, res) {  
 res.send("Hello World!");  
});  
  
var port = process.env.PORT || 3000;  
  
app.listen(port, function() {  
 console.log("listening on #{port}");  
});

13. What is “callback hell” and how can it be avoided?

(Ans). “Callback hell” refers to nested callbacks that have become unreadable or complex to understand.

There are some methods to avoid “calback hell”, which are explained below

1. Use **modularization**. The callbacks are broken out into independent functions which can be called with some parameters.
2. Using Promises

14. How does Node.js support multi-processor platforms, and does it fully utilize all processor resources?

(Ans). As we know Node.js is a **single thread** application, it will run on a single processor. However, Node.js provides support for deployment on multiple-processor systems, to take better and more advantage of the hardware. The Cluster module is one of the core Node.js modules which allow us to running multiple Node.js processes that share the same port. A single instance of Node.js runs in a single thread.To take advantage of multi-core systems the user will sometimes want to launch a cluster of Node.js processes to handle the load. The cluster module allows us to easily create child processes that all share server ports.

15. Explain the CSS “box model” and the layout components that it consists of.

16. Explain what elements will match each of the following CSS selectors:

1. div, p
2. div p
3. div > p
4. div + p
5. div ~ p

(Ans).

|  |  |
| --- | --- |
| **Selector** | **Description** |
| div , p | Selects all <div> elements and all <p> element |
| div p | Selects all <p> elements inside <div> elements |
| div > p | Selects all <p> elements where the parent is a <div> element |
| div + p | Selects all <p> elements that are placed immediately after <div> elements |
| div ~ p | Selects every <p> element that are preceded by a <div> element |